

Beginner's guide to

continuous integration

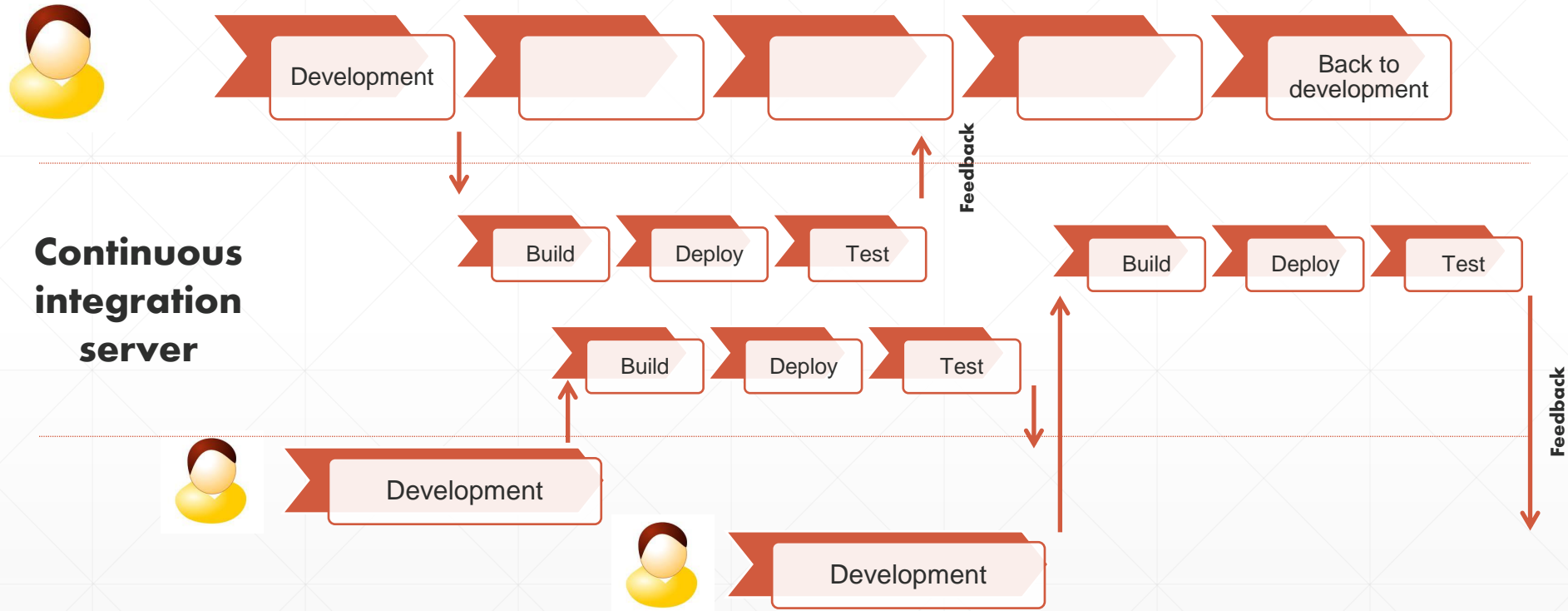
Gilles QUERRET • Riverside Software • US PUG Challenge 2013

What's continuous integration ?

- Build, deployment and tests are long and boring tasks
- Development cycles are shorter, but integration is still long
- Continuous integration is a set of tools and techniques to automate those tasks, and provide notifications as soon as possible



Continuous integration overview



Continuous integration advantages

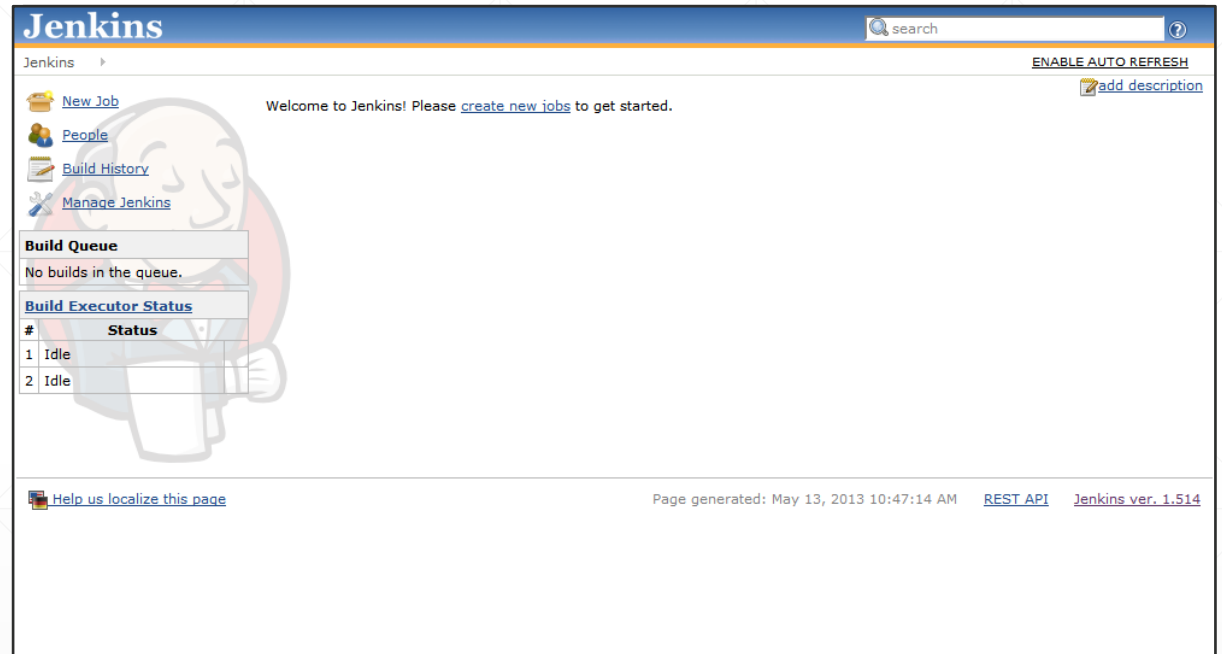
- Fix problems as soon as possible
 - Save valuable time for developers
 - Confidence in what's released
 - Higher project quality
-

Continuous integration process

- Check out from source code repository
 - Build project
 - Execute unit tests
 - Deploy to target system
 - Full tests (optional)
 - Provide feedback
-

Let's start !

- Go to <http://jenkins-ci.org>
- Download Jenkins WAR file
- Execute WAR file
- Go to <http://localhost:8080>



The screenshot shows the Jenkins web interface. At the top, there is a blue header with the 'Jenkins' logo and a search bar. Below the header, there is a navigation menu with links for 'New Job', 'People', 'Build History', and 'Manage Jenkins'. The main content area displays a welcome message: 'Welcome to Jenkins! Please [create new jobs](#) to get started.' There are also links for 'ENABLE AUTO REFRESH' and 'add description'. Below the welcome message, there is a 'Build Queue' section with the text 'No builds in the queue.' and a 'Build Executor Status' table. The table has two columns: '#', 'Status' and two rows of data: '1 Idle' and '2 Idle'. At the bottom of the page, there is a footer with a link to 'Help us localize this page', the text 'Page generated: May 13, 2013 10:47:14 AM', and links for 'REST API' and 'Jenkins ver. 1.514'.

#	Status
1	Idle
2	Idle

A new job in Jenkins

- Create a new job
- Choose a job name and select « Free-style project »
- Then click « OK »

Jenkins

Jenkins > Tous >

[New Job](#)

[People](#)

[Build History](#)

[Manage Jenkins](#)

Job name

Build a free-style software project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Build a maven2/3 project
Build a maven 2/3 project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Build multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Monitor an external job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See [the documentation for more details](#).

Build Queue
No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

[Help us localize this page](#)

Page generated: May 13, 2013 1:26:54 PM [REST API](#) [Jenkins ver. 1.514](#)

New job : source code repository

- Select the source code repository for your project
- CVS and Subversion by default
- Mercurial, Git, Perforce and so on available as plugins

Source Code Management

None

CVS

CVS Projectset

Mercurial

Repository URL ?

Branch ?

Modules ?

Clean Build ?

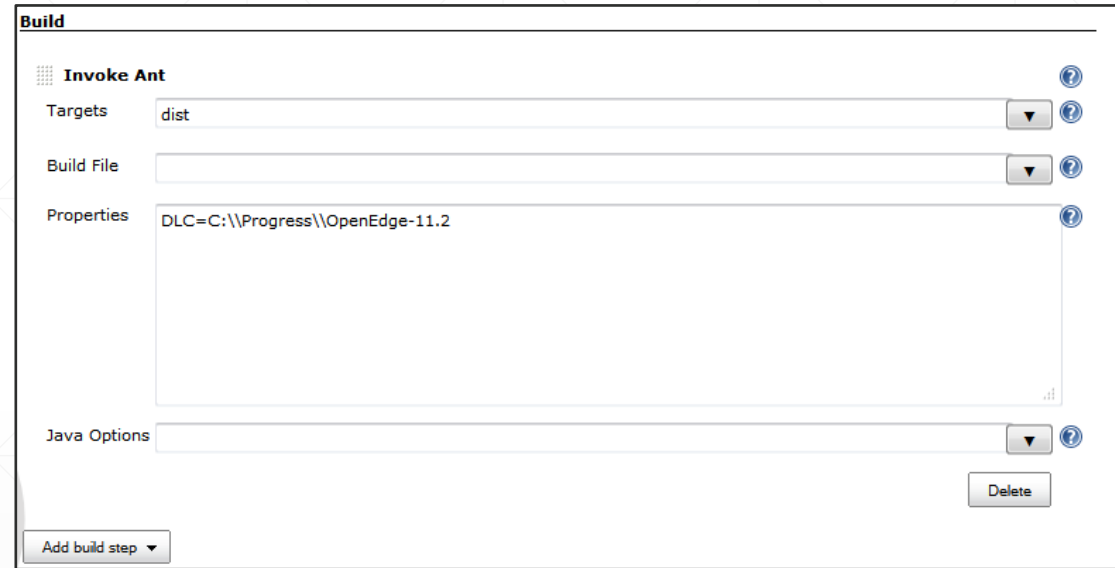
Subdirectory ?

Repository browser ?

Subversion

New job : build steps

- Default build steps are :
 - Windows batch command / Shell script
 - Maven
 - Ant



The screenshot shows a configuration window titled "Build" for an "Invoke Ant" step. The window contains the following fields and controls:

- Invoke Ant**: A header with a grid icon and a help icon.
- Targets**: A dropdown menu with "dist" selected and a help icon.
- Build File**: A dropdown menu with an empty field and a help icon.
- Properties**: A text area containing "DLC=C:\\Progress\\OpenEdge-11.2" and a help icon.
- Java Options**: A dropdown menu with an empty field and a help icon.
- Delete**: A button located at the bottom right of the configuration area.
- Add build step**: A button with a dropdown arrow located at the bottom left of the window.

New job : explanation of existing build steps

- Windows batch commands / Shell scripts :
 - Hard to maintain, not portable, rely on hand-crafted tools for OpenEdge
 - Maven :
 - The new default build system for Java – OpenSource
 - Convention over configuration, dependencies management
 - Ant :
 - The old default build system for Java (but still widely used, and not only for Java) – OpenSource and stable since more than 10 years
 - XML syntax (can be cumbersome), repeatable builds, LOTS of build tasks
-

Using Ant with OpenEdge

- Ant provides a nice structure to add your plugins
 - PCT is an Ant open source plugin for OpenEdge
 - Started almost exactly 10 years ago !
 - Handles the most common OpenEdge related tasks :
 - Database management (create DB, dump/load DF files, dump/load data, ...)
 - Build (r-code from .w/.p.cls, SpeedScript, ...)
 - Libraries (PL management, diff between PL, ...)
 - Procedure execution (with propath, DB connections, options, ...)
-

A sample build script

```
<?xml version="1.0" encoding="utf-8"?>
<project name="MyProject" default="dist">
  <taskdef resource="PCT.properties" />
  <target name="target1">
    <!-- Some build tasks -->
  </target>
  <target name="target2" depends="target1">
    <!-- Some build tasks -->
  </target>
</project>
```

A sample build script

```
<target name="db">  
  <mkdir dir="database/db" />  
  <PCTCreateBase dbName="sports" destDir="database/db"  
    dlcHome="${DLC}" schemaFile="database/dump/sports.df" />  
</target>
```

A sample build script

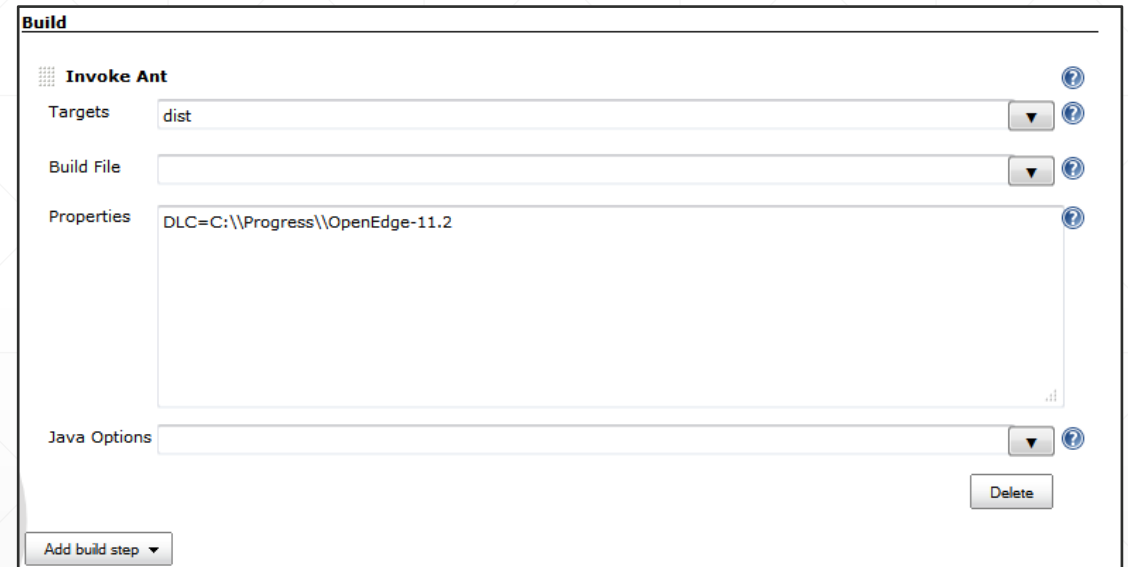
```
<target name="build" depends="db">
  <mkdir dir="build" />
  <PCTCompile destDir="build" dlcHome="${DLC}" >
    <fileset dir="src" includes="**/*.p,**/*.w,**/*.cls" />
    <DBConnection dbName="sports" dbDir="database/db" singleUser="true" />
    <propath>
      <pathelement location="src" />
    </propath>
  </PCTCompile>
</target>
```

A sample build script

```
<target name="dist" depends="build">
  <mkdir dir="dist" />
  <PCTLibrary destFile="dist/sports.pl" baseDir="build" dlcHome="${DLC}" />
  <zip destFile="dist/sports.zip">
    <fileset dir="dist" includes="*.pl" />
    <fileset dir="resources" includes="**/*.jpg" />
  </zip>
</target>
```

Back to Jenkins

- Just add an Ant build step, executing « dist » target.
- Dependencies between tasks will take care of creating database and compiling code
- Build file name is not provided: default is build.xml
- Notice we can pass properties as parameters : multiple jobs can use the same build script with different values



Job configuration page

- Archiving build output
- As a convention, the 'dist' directory contains build artifacts

Post-build Actions

Archive the artifacts

Files to archive

Excludes

Discard all but the last successful/stable artifact to save disk space

Do not fail build if archiving returns nothing

Delete

Add post-build action

And now ?

- Build pipeline
 - Automated tests using the REST adapter
 - Slaves
 - Deployment to virtual machines
 - Manage dependencies
 - Dashboard view
 - Source code analysis
-

Build pipeline

- Dependencies between jobs
- Unit tests are triggered by the main job
- Followed by a deployment to virtual machines

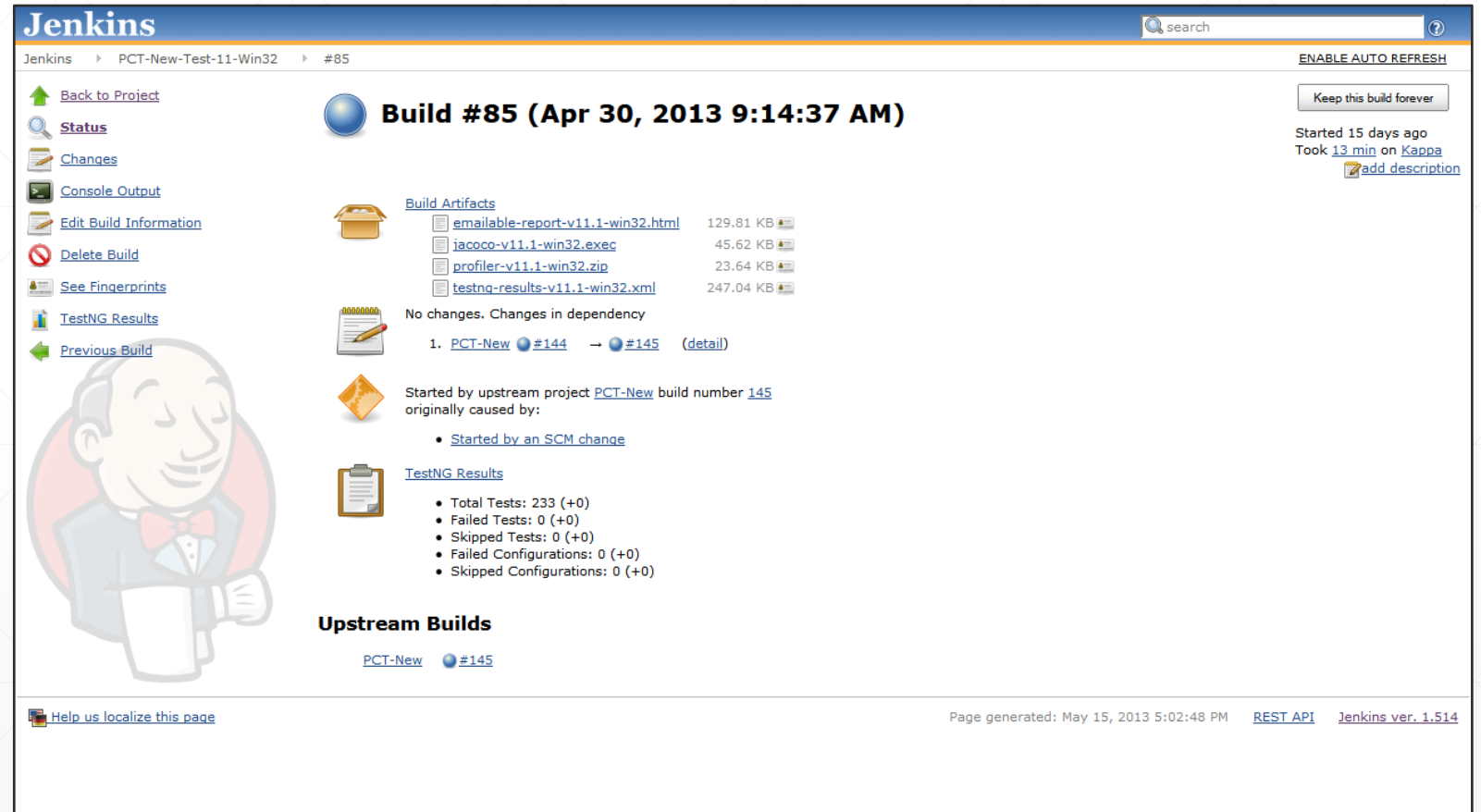
The screenshot displays the Jenkins 'Build Pipeline' view for a pipeline named 'PCT-Dev1'. The interface shows a sequence of jobs connected by arrows, indicating dependencies. The jobs are:

- #149 Dev1-PCT** (green box, 54 sec, May 20, 2013 8:44:47 PM)
- #4 Test1-PCT-10.2...** (yellow box, 16 min, May 20, 2013 8:45:47 PM)
- #4 Test1-PCT-10.2...** (green box, 14 min, May 20, 2013 8:45:47 PM)
- #4 Test1-PCT-10.2...** (green box, 14 min, May 20, 2013 8:45:47 PM)
- #5 Test1-PCT-11.2...** (green box, 12 min, May 20, 2013 9:00:39 PM)
- #3 Test1-PCT-11.2...** (green box, 18 min, May 20, 2013 8:45:47 PM)
- #5 Test1-PCT-9.1E...** (yellow box, 8 min 38 sec, May 20, 2013 9:00:39 PM)
- Test1-PCT-Join** (blue box)
- Test1-PCT-Join** (blue box)
- Test1-PCT-Join** (blue box)
- Test1-PCT-Join** (blue box)
- Test1-PCT-Join** (blue box)
- Test1-PCT-Join** (blue box)

The Jenkins interface includes a search bar, user 'gquerret', and 'log out' link. The page footer shows 'Page generated: May 23, 2013 7:24:10 AM REST API Jenkins ver. 1.515'.

Automated tests

- Tests are executed just after the build
- Under Win32, OE 11.2, but different configurations are being used
- See Mike Fechner package for unit tests
- Or have a look at Prounit
- Or use API on appservers, with OpenAPI or REST adapter



The screenshot displays the Jenkins web interface for a specific build. The page title is "Jenkins" and the breadcrumb navigation shows "Jenkins > PCT-New-Test-11-Win32 > #85". The main heading is "Build #85 (Apr 30, 2013 9:14:37 AM)". On the left sidebar, there are links for "Back to Project", "Status", "Changes", "Console Output", "Edit Build Information", "Delete Build", "See Fingerprints", "TestNG Results", and "Previous Build". The main content area shows "Build Artifacts" with a table of files: "emailable-report-v11.1-win32.html" (129.81 KB), "jacoco-v11.1-win32.exec" (45.62 KB), "profiler-v11.1-win32.zip" (23.64 KB), and "testng-results-v11.1-win32.xml" (247.04 KB). Below this, it states "No changes. Changes in dependency" and lists a dependency on "PCT-New #144" leading to "PCT-New #145 (detail)". A note indicates the build was "Started by upstream project PCT-New build number 145 originally caused by: Started by an SCM change". The "TestNG Results" section shows: "Total Tests: 233 (+0)", "Failed Tests: 0 (+0)", "Skipped Tests: 0 (+0)", "Failed Configurations: 0 (+0)", and "Skipped Configurations: 0 (+0)". The "Upstream Builds" section lists "PCT-New #145". The footer includes "Help us localize this page", "Page generated: May 15, 2013 5:02:48 PM", "REST API", and "Jenkins ver. 1.514".

Slaves

- Public Jenkins instance for Apache Software foundation manages more than 300 jobs
 - With so many jobs, you need to dispatch them on different machines
 - Jenkins provides a simple yet powerful master/slave mechanism, where jobs can be assigned to slaves, while the UI (configuration and reports) are still on the master server
 - Extremely useful not only to dispatch load, but also to test on different configurations
-

Deploy to virtual machines

- Snapshot is your best friend
 - Define snapshots for various configuration (OpenEdge, OS, and other requirements)
 - When a build is completed, use the VIX (or EC2) API to :
 - Restore to snapshot
 - Copy artifacts to virtual machine
 - Execute your installer (or upgrade process)
 - Pause your virtual machine (or not !)
-

Manage dependencies

- Keep your builds small
 - Build artifacts can be kept in Jenkins, or uploaded to an artifact repository
 - Grab dependencies when building downstream jobs (copy artifacts Jenkins plugin or download from artifact repository)
 - During development, download them from Jenkins or artifact repository
 - Define requirements : OpenEdge, Sonic, ... which shouldn't be managed by the repository
-

Dashboard view

CppTools #572	DeleteOldPipelineData #783	Grouper-calibration #664	Grouper-demonstrator #1293	Grouper-event-detection-vhdl-test #65	Grouper-performance-events-aeroflex #50
Grouper-performance-events #117	Grouper-performance-test #485	Grouper-rpm-release #14	Grouper-rpm-tag #162	Grouper-rpm #580	Grouper-selenium #363
Grouper-update-test #139	GrouperDevserve #659	GrouperOnBlade-all_recipies #573	GrouperOnBlade-memleak_system_test #645	GrouperProd #858	GrouperTrunk-event_data_system_test #1057
GrouperTrunk-multiplex_system_test #385	GrouperTrunk-raw_data_system_test #1045	GrouperTrunk-system_test #2300	GrouperTrunk-system_test_aeroflex #544	GrouperTrunk-weekend-test #38	GrouperTrunk #3632
Grouper_ADLINK_all_sample_rates #341	PaddleFish-rpm #386	PaddleFish #395	SampleCPP #405	SystemBoard #708	barracuda-lite-prerelease-2.0 #341
barracuda-lite-trunk #331	barracuda-prerelease-71.0 #46	barracuda-trunk #1082	barramundi-prerelease-1.0 #702	blackfin-prerelease-5.0 #46	blackfin-trunk #66
conger-peakdetector-system-test #237	conger-pipeline-trunk-memleak #683	conger-pipeline-trunk #1650	conger-rpm-tag #56	conger-trunk #912	fastStools #1376
grouper-api-prerelease-0.5 #331	grouper-api-trunk #493	moray-pipeline-trunk #415	moray-prerelease-10.0 #1	ont-hdf5-prerelease-1.0 #26	stickleback-prerelease-018 #543
stickleback-trunk #562	swordfish-prerelease-3.0 #128	swordfish-stable #105			

Source code analysis

Navigation: Dashboards | Projects | Measures | Reviews | Settings | Log in | Search

Project: PCT - OpenEdge module

Version 145 - 30 Apr 2013 10:17 | Time changes...

Lines of code 12,157 18,900 lines	Files 66 13 directories
Comments 27.6% 4,628 lines	Duplications 0.0% 0 lines 0 blocks 0 files

Violations 29

Blocker	0
Critical	0
Major	14
Minor	0
Info	15

Rules compliance 99.7%

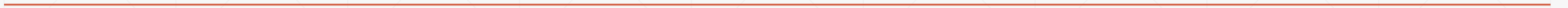
Events All

30 Apr 2013	Profile	OpenEdge Sonar version 2
30 Apr 2013	Version	145
22 Apr 2013	Version	143
03 Apr 2013	Version	142

Key: eu.rssw.pct.oe:PCT-New
Language: OpenEdge
Profile: OpenEdge Sonar (version 2)
Alerts: [RSS Feed](#)

Powered by [SonarSource](#) - Open Source [LGPL](#) - v.3.5 - [Plugins](#) - [Documentation](#) - [Ask a question](#)

Questions ?



References

- Ant : <http://ant.apache.org>
 - PCT : <http://code.google.com/p/pct>
 - Jenkins : <http://jenkins-ci.org>
-